Is the Activity Normal? If Not, What Type of Suspicious?

Lakhyadeep Konwar[1, *, a], Navajit Saikia[2, b] and Subhash Chandra Rajbongshi[3, a]

[a]*Dept. of ECE, Gauhati University, Assam, India, 781014*
[b]*Dept. of ETE, Assam Engineering College, Assam, India, 781013*

Abstract: *Human activity recognition in videos is a major research area due to advancements in automatic visual surveillance system design. We need to design a learning-based AI system that uses CCTV cameras connected to a data center, a well-trained algorithm and an automatic indication system to classify both normal and unusual activities. In this paper, we modified ResNet 10 architecture and achieve classification accuracy of 98.66% for UU2 dataset, 53.68% for UCF Crime 13 dataset, and 79.65% for UEDM dataset which is better as compared to the original ResNet 10 model.*

Keywords: **Activity Recognition, Deep Learning, Machine Learning.**

1. INTRODUCTION

Crime and terrorist activities are increasing, posing challenges for law enforcement. Deploying troops and using CCTV cameras in various locations has limitations. To address this, an algorithm capable of detecting unusual activities and alerting authorities is needed to apprehend criminals in a timely manner and prevent further crimes.

In public places, CCTV cameras provide security and surveillance. While there are technologies to detect abnormal activities, automated surveillance cameras that can identify suspicious activities are not widely used. One challenge is differentiating between normal and suspicious activities, such as recognizing the intent behind someone holding a gun. The task involves classifying activities using computer vision and machine learning. Deep learning (DL) techniques offer better performance but require high computational power. 3D convolutional neural network (3DCNN) is advantageous for analyzing video data and has become popular for activity classification tasks.

Residual Network (ResNet) is a deep neural network (DNN)architecture designed to address the vanishing gradient problem in deep networks. It incorporates skip connections or shortcuts to enable the gradient to flow directly through the network, bypassing certain layers. ResNet has achieved state-of-the-art performance in various computer vision tasks and has been extended to 3D for video classification tasks.

In neural network architectures, an activation function determines whether a neuron should be activated by calculating the weighted sum, adding bias and producing output from a set of input values fed to a node. It is essential to carefully consider an activation function to achieve a better fit for the model and attain a higher level of accuracy.

The following are the primary contributions of this paper:

- We created the Usual Unusual 2 Dataset (UU2) to classify usual and unusual activities in daily life. Data was collected from the UCF101 dataset for usual activities and the UCF Crime dataset for unusual activities.
- We introduced the UCF Crime 13 dataset, which includes 13 anomaly classes from the original UCF Crime dataset of 14 classes.
- We have compiled a new dataset called Usual Event Dataset from Multimedia (UEDM) from platforms like YouTube, Facebook and Instagram.
- We've modified ResNet 10 with variants such as ResNet 10_2 through ResNet 10_8, which will be discussed in Section 4.

The rest of the paper is organized as follows. A brief literature review of anomaly as well as unusual activity classification is presented in Section 2. Section 3 discussed

theoretical aspects of ResNet and activation functions related to this work. Section 4 elaborates the proposed work followed by experimental result section in Section 5 and finally Section 6 provides conclusion and feature directions of this work.

2. RELATED WORK

Different researchers perform various tasks for detecting anomalies or unusual events in visual surveillance systems. Some of them are discussed below.

Freer et al. [1] proposed a poster-based algorithm for object detection in CCTV images to determine if an intruder is present. Maurin et al. [2] proposed a vision-based system aimed at enhancing traffic management in urban areas, such as urban intersections, and streets after athletic events. The system can effectively detect and track both pedestrians and vehicles. To detect pedestrians and vehicles, the system employs optical flow and background removal algorithms. Finally, it tracks the detected objects using the Kalman filter. Mahadevan et al. [3] proposed a framework for detecting anomalies in crowded environments. The framework is based on a mixture of dynamic textures and outlines labeled as anomalies. In their study, Roshtkhari et al. [5] introduced a new method for analyzing videos and learning about both common and unusual behaviors that occur in surveillance footage. This technique allows for the creation of a hierarchical model of the scene that can help researchers better comprehend different behaviors. Lung et al. [6] proposed a spatio-temporal descriptor (STD) for abnormal human activity recognition. It is based on the spatio-temporal feature of an image. Anomaly detection systems are not very effective due to their high false-positive rate and lack of real-time capabilities. Moreover, their feature selection techniques limit their usage to specific cases. To address these issues, Narasimhan et al. [8] proposed a dynamic anomaly detection and localization system that utilizes sparse denoising autoencoder to learn relevant features automatically. This reduces the computational time and enhances the system's effectiveness. In their work, Wang et al. [10] introduced the abnormal event detection network (AED-NET) to identify unusual events in a crowded environment. They employed PCA-NET to extract high-level features of the crowd's situation and then used kPCA to detect anomalies within the scene. To overcome the problem of overfitting, the authors added a local response normalization (LRN) layer to the original AED-NET. Dong et al. [11] proposed a dual discriminator based GAN structure for video anomaly detection. Lv et al. [12] proposed a method called weakly supervised anomaly localization (WSAL) which focuses on localizing anomalous segments within anomalous videos over time. The method uses a high-order context encoding model to extract semantic representations and measure dynamic variations. Cai et al. [14] proposed an appearance motion memory consistency network (AMMC-Net) for identifying abnormal frames in multimodal signals, to detect abnormal events. In their study, Hu et al. [15] proposed a self-attention prototype unit (APU) that can encode the ordinary latent space as prototypes in real-time, without any extra memory cost. They also introduced a circulative attention mechanism as a backbone to create a new feature extracting learner called circulative attention unit (CAU). This unit enables fast adaptation capability on new scenes by using a few iterations of the update. Sarker et al. [16] proposed a method for detecting anomalies in video surveillance scenes using a weakly supervised learning algorithm. They extracted spatio-temporal features from each surveillance video by using a temporal convolutional 3D neural network (T-C3D). The authors introduced a unique ranking loss function to increase the difference between the classification scores of anomalous and normal videos by reducing the number of false negatives. Ullah et al. [17] have proposed an efficient and lightweight convolutional neural network (CNN) based framework for recognizing anomalies in surveillance environments. The framework reduces time complexity by extracting spatial CNN features from a series of video frames. These features are then fed into the proposed residual attention-based long short-term memory (LSTM) network, which can accurately detect anomalous activity in surveillance videos.

Habib et al. [18] developed a lightweight method that can accurately detect violent activities in surveillance environments. The proposed system generates real-time alarms to notify law enforcement agencies in case of any violent activity or accidents, which can help them take appropriate action to avoid accidents and stampedes. In their research, Hussain et al. [20] proposed a framework that prioritizes cameras in a large surveillance network based on feedback from an activity recognition system. The framework aims to conserve energy by intelligently adjusting the priority of cameras. To identify salient frames, the researchers used a frame differencing method on the online video stream. They then employed a lightweight 3DCNN architecture to extract spatio-temporal features from the selected frames. Finally, the probabilities predicted by the 3DCNN network and the metadata of the cameras are processed using a linear threshold gate sigmoid mechanism to control the priority of the camera.

Following section describes the theory related to the proposed model.

3. THEORETICAL CONSIDERATIONS

3DCNN, shown in *Figure* 1, has gained popularity in video classification due to its ability to analyze object positions over time. It creates a 3D activation map in the convolutional operation, aiding in understanding time and volumetric context. A 3D filter is applied to perform 3D convolution by moving the kernel in the x, y and z directions.

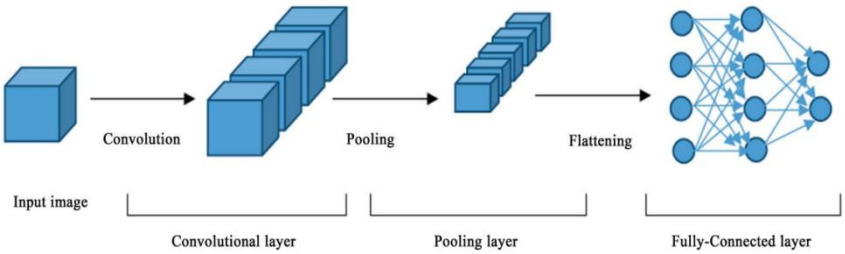

Figure 1. Basic architecture of a 3DCNN [13].

Here, the value of each position in the feature maps of the layer is presented as follows.

$$v_{ij}^{xyz} = tanh\left(b_{ij} + \sum_{k}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijk}^{xyz} v_{(i-1)k}^{(x+p)(y+q)(z+r)}\right) \quad (1)$$

Where, w_{ijk}^{xyz} represents the value of the kernel connected to the feature map in the previous layer.

The feature map output is in a 3D volumetric space. To achieve 3D convolution, we wind around the center of the cube and stack adjacent layers on top of each other. By interconnecting the feature maps, we can capture motion information. The 3D max pooling layer is a form of non-linear down-sampling of an input tensor. It partitions the input tensor data into 3D sub-tensors into the three dimensions and then selects the element of each sub-tensor with the maximum numeric value. At the final stage, the max-pooling converts the input tensors to output tensors by replacing each of the sub-tensors with its maximum elements. After features are extracted, we need to classify the data into various classes. To do this, we can use the fully connected layer, which is the final stage of a CNN for end-to-end training of a model.

3.1. ResNet

Some systems are harder to optimize than others, leading to a decrease in training accuracy [7]. While deeper models shouldn't have a higher training error than shallow ones, constructing such models isn't always possible. ResNet [7] is a solution that uses shortcut connections to turn the network into its residual version, as seen in *Figure* 2.

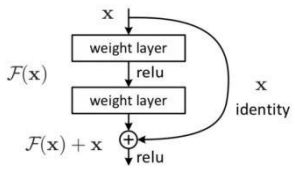

Figure 2. Residual learning block of ResNet [7].

The building block can be defined as:

$$y = F\left(x, \{W_i\}\right) + x \tag{2}$$

Where, x and y are input and output vectors of the layers. The function $F\left(x, \{W_i\}\right)$ represents the residual mapping to be learned. The dimensions of x and F must be the same. If they are not, for instance, when the input/output channels are changed, linear projections can be created using shortcut connections to match the required dimension is given by following equation.

$$y = F\left(x, \{W_i\}\right) + W_s x \tag{3}$$

Where, W_s be the linear projection. *Equation* 2 can be used when inputs and outputs are of the same size. With increasing dimensions, two options are available: (a) the shortcut performs identity mapping and (b) the projection shortcut in *Equation* 3 is used to match dimensions with a 1 × 1 convolution. In both cases, the shortcut goes across feature maps of two sizes with a stride of 2.

3.2. Activation Function

The activation function in an artificial neural network (ANN) calculates the node's output based on its input and weights, introducing non-linearity to represent complex patterns in data. Without non-linearity, the network would behave like a linear regression model. The activation function decides whether a neuron should be activated by calculating the weighted sum and adding bias. In this work we have used rectified linear unit (ReLU) and LeakyReLU activation function on ResNet. Therefore, in the following we will discuss about them.

3.2.1. ReLU: ReLU is a commonly used activation function in neural networks. It returns the input if it's positive and 0 if it's negative. This function is popular for its simplicity and effectiveness in preventing the vanishing gradient problem. It has a derivative function for propagation and offers less time and complexity. The mathematical definition of ReLU is shown below.

$$f(x) = \begin{cases} x, & if\ x \geq 0; \\ 0, & else. \end{cases} \tag{4}$$

In short, if the input is positive, the output is the same as the input; if the input is negative, the output is 0. ReLU helps prevent the vanishing gradient problem seen with other activation functions like sigmoid or tanh.

3.2.2. LeakyReLU: Leaky ReLU is a variant of the ReLU activation function used in deep learning models. It is similar to ReLU but has a small slope for negative input values as derived by following equation.

$$f(x,\alpha) = \begin{cases} x, & if\ x \geq 0 \\ \alpha x, & else \end{cases} \quad (5)$$

Where, x is the input to the function, α is a small positive value (typically 0.01) and $f(x)$ is the output.

LeakyReLU, like ReLU, prevents the "dying ReLU" problem by ensuring that neurons in the network don't always output zero. This helps the network learn more effectively.

ResNet can be customized with various parameters such as activation function, model depth size and connections. The following section outlines the proposed work.

4. PROPOSED WORK

Due to the availability of higher computational resources, deeper as well as wider architecture can be used for different object classification task, which provides better performances. It may be noted here that, due to the availability of higher computational resources, we can modify deeper architecture of DNN model to achieved a better result. Hence, this work proposes to consider original ResNet 10, ResNet 10_2, ResNet 10_3, ResNet 10_4, ResNet 10_5, ResNet 10_6, ResNet 10_7 and ResNet 10_8 models with both of the connections A and B. Therefore, a total of 16 models are trained and tested for each of the three different datasets (UU2, UCF Crime 13 and UEDM) and obtained results are provided in the result and discussion section.

The overall implementation for this work is carried out with Python 3.10.4 in PyTorch framework; where, "Numpy" and "OpenCV" is used for data manipulation and video processing. *Figure* 3 shows the basic block diagram of the proposed method. The proposed method has mainly four parts: input video section, data augmentation section, network selection section and output class. Here, each of the video clip is resized to 112×112 before fading it to the network. The output classes are named as class 1, class 2, class 3, ..., class N.

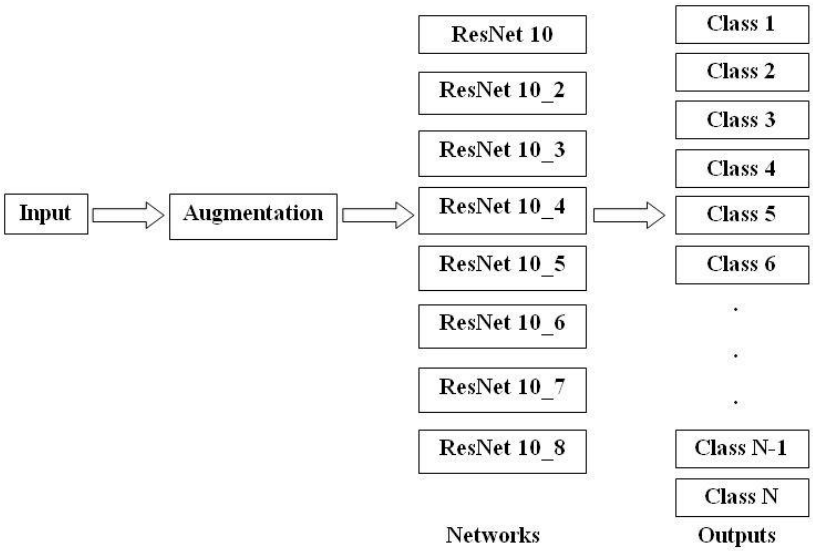

Figure 3. Basic block diagram of proposed activity classification system.

4.1. ResNet 10 varients

In ResNet 10 original, we have used the original version of 3D ResNet 10. We have change the kaiming function in ResNet 10_2 model. We use LeakyReLu activation function on the original model of ResNet 10 in ResNet 10_3 and LeakyReLu activation function on ResNet 10_2 model in ResNet 10_4. The basic block diagram of ResNet 10, ResNet 10_2, ResNet 10_3 and ResNet 10_4 is shown in *Figure* 4. In the ResNet 10_5 and ResNet 10_6 model, we have added a convolutional layer over ResNet 10_2 model for ReLU and LeakyReLU activation function as shown in *Figure* 5. Similarly we add an another convolutional layer on ResNet 10_5 and ResNet 10 6_and rename the models as ResNet 10_7 and ResNet 10_8 as shown in *Figure* 6.

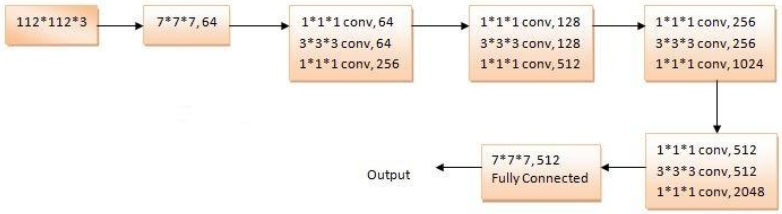

Figure 4. Basic block diagram of ResNet 10, ResNet 10_2, ResNet 10_3 and ResNet 10_4.

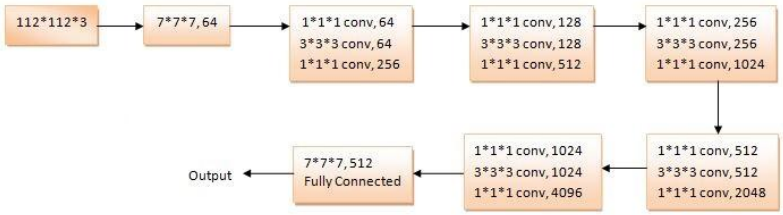

Figure 5. Basic block diagram of ResNet 10_5 and ResNet 10_6.

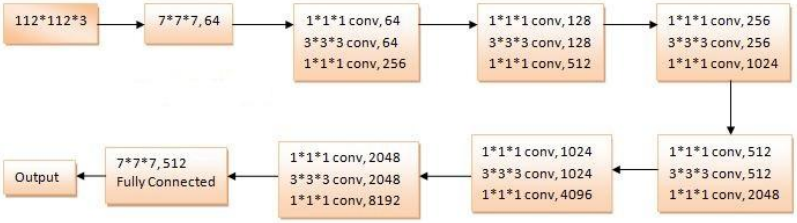

Figure 6. Basic block diagram of ResNet 10_7 and ResNet 10_8.

4.2. Dataset

Below, we have discussed about three different activity datasets: UU2, UCF Crime 13 and UEDM are used for our work.

4.2.1. UU2 Dataset: We consider two classes for usual and unusual activity classification problem; where, we use 13 classes of actions e.g. apply eye makeup, apply lipstick, baby crawling, band marching, blow dry hair, blowing candles, brushing teeth, haircut, mopping floor, playing violin, push ups, typing and writing on board from UCF101 [4] dataset for usual activities and 13 classes e.g. abuse, arrest, arson, assault, burglary, explosion, fighting, road accidents, robbery, shooting, shoplifting, stealing and vandalism from UCF Crimes [9] as unusual or criminal activities. This dataset contains a total of 2597 videos. Out of these, 950 videos belong to the unusual class, while the remaining 1647 videos belong to the usual class. *Figure* 7 shows a visual representation of some

activities of the videos belonging to the (a) usual and (b) unusual activities of the UU 2 dataset.

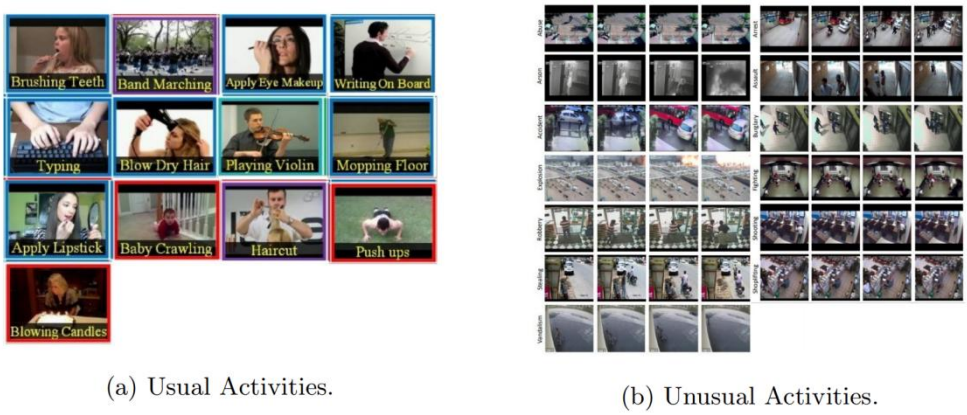

Figure 7: Visual representation of different activities of Usual Unusual 2 dataset.

4.2.2. UCF Crime 13: The UCF Crime 13 dataset consists of a total of 950 real world surveillance videos that are long and untrimmed. This dataset includes 13 different types of anomaly activities i.e. abuse, arrest, arson, assault, road accidents, burglary, explosion, fighting, robbery, shooting, stealing, shoplifting and vandalism. For our purposes, we have only used the 13 anomaly classes and ignored the normal class from UCF Crime dataset [9]. *Figure* 8 shows visual representation of different activities of the UCF Crime 13 dataset.

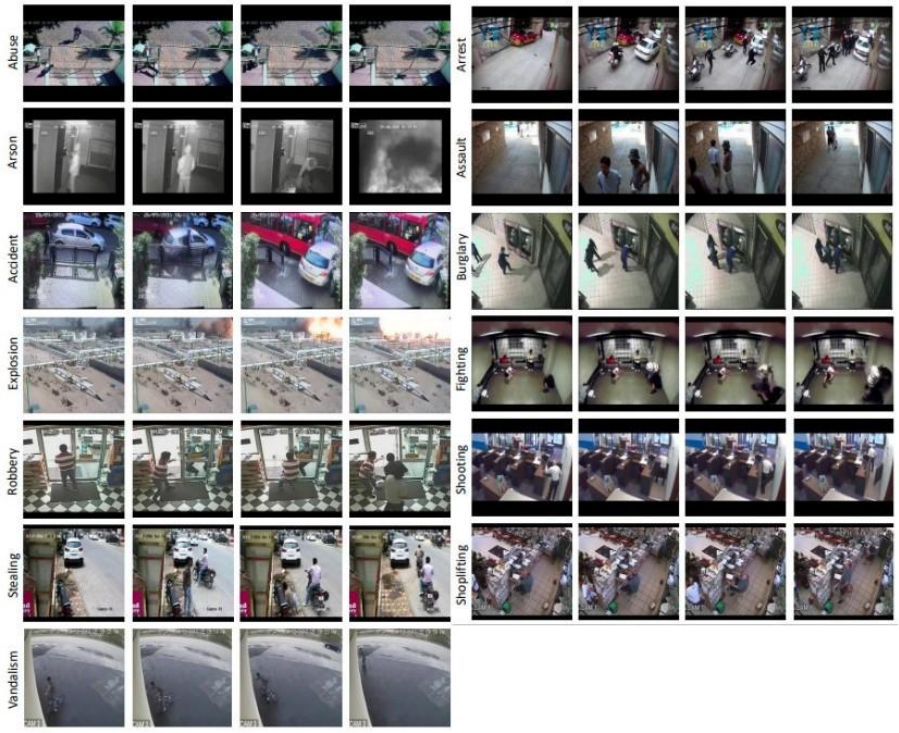

Figure 8: Visual representation of different activities of UCF Crime 13 dataset.

4.2.3. UEDM: The UEDM is a collection of videos showcasing unusual human activities, categorized into five classes: bag snatching, gun pointing and shooting, kicking, knife attack and punching. The dataset comprises 528 video clips with a total of 1406 random participants. The videos capture performers from various angles and durations, presenting challenges such as different lighting conditions and activities being performed in various environments, including inside moving vehicles. *Figure* 9 shows visual representations of

the activities performed by different individuals in different video streams and frames, with eight different images displayed for each class.

(a) Bag Snatching. (b) Gun Pointing and Shooting.

(c) Knife Attack. (d) Kick.

(e) Punching.

Figure 9: Visual representation of different activities of UEDM dataset.

4.3. System Used

We used Dell Precision T7610 Workstation of 64 GB RAM, 8 TB of HDD and NVIDIA GeForce 2080 Ti RTX graphics card to improve the computation speed. Here, we used Ubuntu 18.04 as our operating system.

4.4. Data Preparation

First of all, convert all of those video files to frames with a frame rate of 25 frames per second (FPS) for all of the available classes and then create annotated file for all of those videos in accordance with their classes. Here, we used .json file format to annotate data. We follow the format of UCF 101 dataset annotation and prepare our dataset for training, validation and testing process. After data annotation, augmentation is necessary before fed data to different ResNet 10 models for both training and testing of datasets. Here, all the datasets is split as 80% for training+validation and 20% for testing. The annotated data are used to study performances of the selected variants of ResNet 10 architecture.

4.5. Details of the Implementation

In our implementation, we have divided all of the training and testing videos into 32 non-overlapping video segments before performing the feature extraction task. Then each video frame is resized into 112 × 112 pixel size.

4.5.1. Augmentation: The study separately considers temporal and spatial augmentation to achieve spatio-temporal augmentation. The input clip to the network consists of 16 frames. For temporal augmentation, uniform sampling is used to select temporal positions, and input clips are derived by considering frames around these positions. For spatial augmentation, a random spatial position is selected in each input clip, and multi-scale cropping is carried out with a spatial scale from $\{1, \frac{1}{2^{\frac{1}{4}}}, \frac{1}{2^{\frac{3}{4}}}, \frac{1}{2}\}$.

The resulting clips are horizontally flipped with a 50% probability and resized to 3×16×112×112. They are then labeled with the class of the original video.

4.5.2. Learning: In our implementation, we use the stochastic gradient descent (SGD) optimizer with a cross-entropy loss function. The training parameters include a damping ratio of 0.9, momentum of 0.9, weight decay of $1e^{-1}$ and dropout of 0.2. We use a batch size of 32, learning rate of 0.01 and train for 250 epochs. Upon converging validation loss,

we reduce the learning rate by a factor of 10^{-1} three times. For instance, if the initial learning rate is 0.01, it is adjusted to 0.001 and further decreased to 0.0001 and 0.00001 after the 40th and 55th epoch for fine-tuning.

4.5.3. Network: We use 3D architectures of ResNet 10, ResNet 10_2, ResNet 10_3, ResNet 10_4, ResNet 10_5, ResNet 10_6, ResNet 10_7 and ResNet 10_8 for both of the connections A and B, which results in 16 network variations. These models are trained and tested for RGB modality.

Following section presents the experimental results and performance analysis of the proposed methods.

5. EXPERIMENTAL RESULTS

All the 16 variants of ResNet 10 network architectures as derived above are trained and tested separately for UU2, UCF Crime 13 and UEDM dataset. For all 16 variants of network architectures we consider learning rate of 0.01. Batch size is considered here is 32 for all of the experimentation's. In the following we present the performances of all the networks with respect to the datasets we have used.

5.1. UU2 Dataset

Considering UU2 dataset with input size of 112 × 112 and all other network parameter, altogether 16 variants of ResNet 10 architecture are trained and tested for studying their performances. *Table* 1 summarizes the classification rates and associated network complexities for ResNet 10. The best classification accuracy of 98.66% is found with ResNet 10_7_A network which has 377.54 M number of parameters. The model performs around 1.33% better than the other top performing model in the table. It uses 4.57 GB and 4.02 GB of GPU memory during training and testing respectively. Here, the model required epoch time of 33.35 seconds (per epoch) and batch time of 0.5131 seconds. The test time required for this model is about 178 seconds. The system required 1638 MB of system memory to store the trained model which is higher than most of the model that we have tested. *Figure* 10 (a) and (b) shows the plot of epoch vs train loss for UU2 dataset of different ResNet 10 architecture with shortcut A and B. *Figure* 10 (c) and (d) shows the plot of epoch vs test loss for UU2 dataset of different ResNet 10 architecture with shortcut A and B.

Table 1. Comparison table for the different ResNet 10 model architectures in terms of accuracy (A) in %, number of parameters (NP in millions), GPU memory required for training (GTR in GB), epoch time (ET in seconds), batch time (BT in seconds), GPU memory required for testing (GTS in GB), test time (TT in seconds) and model size (MS in MB) against each of the model using UU2 dataset.

Model	A	NP	GTR	ET	BT	GTS	TT	MS
ResNet 10 A	96.16	14.23	2.51	18.10	0.2785	1.69	128	113.9
ResNet 10 B	96.55	14.40	2.53	18.17	0.2795	1.71	124	115.3
ResNet 10_2 A	96.31	14.23	2.51	21.04	0.3237	1.69	127	113.9
ResNet 10_2 B	96.12	14.40	2.53	21.26	0.3271	1.71	126	115.3
ResNet 10_3 A	95.20	14.23	2.51	21.03	0.3235	1.69	128	113.9
ResNet 10_3 B	96.93	14.40	2.53	21.76	0.3348	1.71	124	115.3
ResNet 10_4 A	95.78	14.23	2.51	20.93	0.3220	1.69	128	113.9
ResNet 10_4 B	96.50	14.40	2.53	21.04	0.3237	1.71	125	115.3
ResNet 10_5 A	97.30	94.41	2.97	25.05	0.3854	2.46	141	472.3
ResNet 10_5 B	97.32	95.11	2.99	26.01	0.4002	2.48	135	475.8
ResNet 10_6 A	96.71	94.41	2.97	27.01	0.4155	2.46	142	472.3
ResNet 10_6 B	96.55	95.11	2.99	28.14	0.4329	2.48	136	475.8
ResNet 10_7 A	**98.66**	**377.54**	**4.57**	**33.35**	**0.5131**	**4.02**	**178**	**1638**
ResNet 10_7 B	96.93	380.34	4.61	33.82	0.5203	4.05	169	1659
ResNet 10_8 A	97.11	377.54	4.57	34.83	0.5358	4.02	178	1638
ResNet 10_8 B	97.33	380.34	4.61	35.15	0.5408	4.05	174	1659

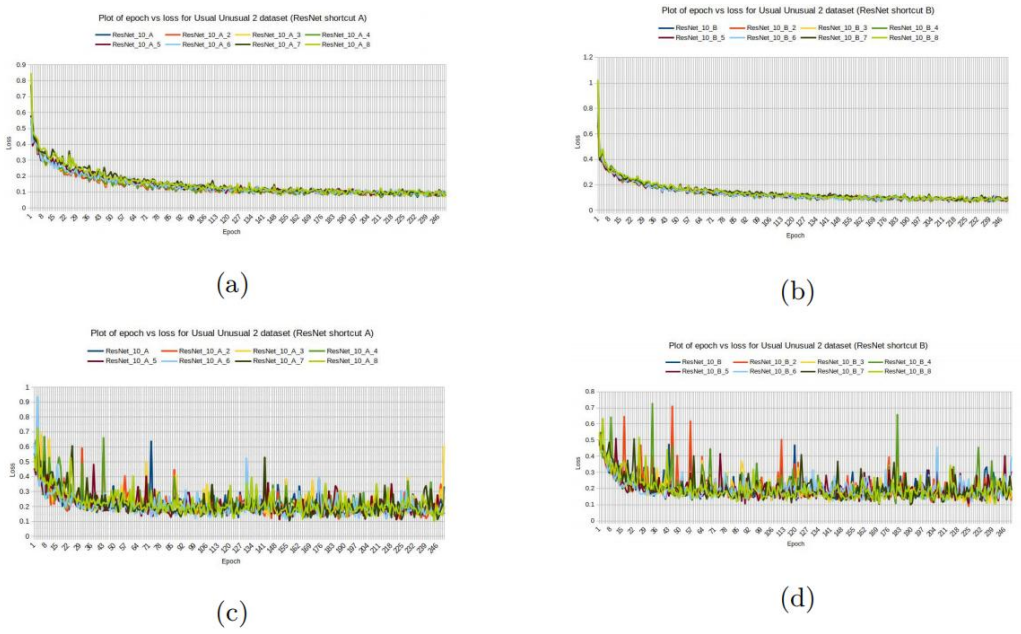

Figure 10: Plot of (a). epoch vs train loss for UU2 dataset with ResNet 10_A, (b). epoch vs train loss for UU2 dataset with ResNet 10_B, (c). epoch vs test loss for UU2 dataset with ResNet 10_A and (d). epoch vs test loss for UU2 dataset with ResNet 10_B.

5.2. UCF Crime 13 Dataset

The observed classification accuracy for 16 different variants of ResNet 10 is shown in *Table* 2. The best performance for UCF Crime 13 dataset is found to be 53.68% for ResNet 10_7_A where 377.54 M parameters are involved. The GPU memory used here is 4.57 GB and 4.02 GB for training and testing respectively. Here, epoch time and batch time required is about 27.09 and 1.1288 second. It required testing time of 184 seconds and model size is about 1638 MB. *Figure* 11 (a) and (b) shows the plot of epoch vs train loss for UCF Crime 13 dataset of different ResNet 10 architecture with shortcut A and B. *Figure* 11 (c) and (d) shows the plot of epoch vs test loss for UCF Crime 13 dataset of different ResNet 10 architecture with shortcut A and B.

Table 2. Comparison table for the different ResNet 10 model architectures in terms of accuracy (A) in %, number of parameters (NP in millions), GPU memory required for training (GTR in GB), epoch time (ET in seconds), batch time (BT in seconds), GPU memory required for testing (GTS in GB), test time (TT in seconds) and model size (MS in MB) against each of the model using UCF Crime 13 dataset.

Model	A	NP	GTR	ET	BT	GTS	TT	MS
ResNet 10 A	49.47	14.23	2.51	9.38	0.3908	1.69	124	113.9
ResNet 10 B	48.95	14.40	2.53	9.43	0.3929	1.71	120	115.3
ResNet 10_2 A	51.11	14.23	2.51	9.49	0.3954	1.69	125	113.9
ResNet 10_2 B	48.42	14.40	2.53	10.24	0.4267	1.71	121	115.3
ResNet 10_3 A	51.10	14.23	2.51	10.14	0.4225	1.69	123	113.9
ResNet 10_3 B	49.47	14.40	2.53	10.92	0.4550	1.71	120	115.3
ResNet 10_4 A	41.58	14.23	2.51	10.59	0.4413	1.69	123	113.9
ResNet 10_4 B	50.10	14.40	2.53	11.40	0.4750	1.71	120	115.3
ResNet 10_5 A	50.00	94.41	2.97	13.05	0.5438	2.46	133	472.3
ResNet 10_5 B	52.63	95.11	2.99	15.15	0.6313	2.48	129	475.8
ResNet 10_6 A	52.02	94.41	2.97	14.25	0.5938	2.46	135	472.3
ResNet 10_6 B	46.31	95.11	2.99	16.09	0.6704	2.48	131	475.8
ResNet 10_7 A	**53.68**	**377.54**	**4.57**	**27.09**	**1.1288**	**4.02**	**184**	**1638**
ResNet 10_7 B	50.53	380.34	4.61	28.17	1.1738	4.05	180	1659
ResNet 10_8 A	48.42	377.54	4.57	27.17	1.1321	4.02	186	1638
ResNet 10_8 B	45.26	380.34	4.61	27.84	1.1600	4.05	182	1659

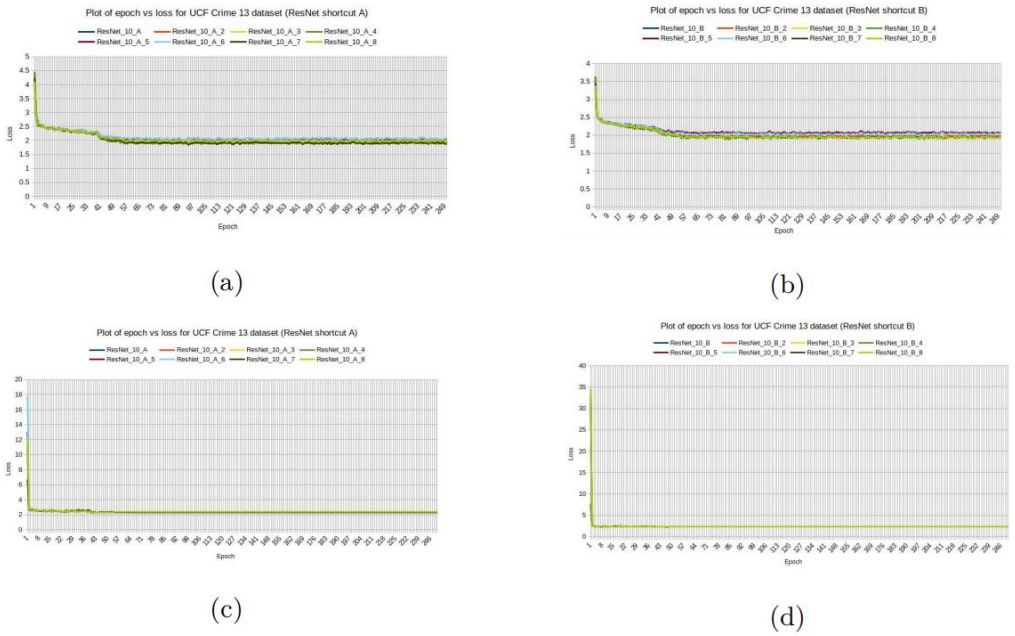

Figure 11: Plot of (a). epoch vs train loss for UCF Crime 13 dataset with ResNet 10_A, (b). epoch vs train loss for UCF Crime 13 dataset dataset with ResNet 10_B, (c). epoch vs test loss for UCF Crime 13 dataset with ResNet 10_A and (d). epoch vs test loss for UCF Crime 13 dataset with ResNet 10_B.

5.3. UEDM

The observed performances of the 16 variants of ResNet 10 are shown in *Table* 3. The best performance for ResNet 10 variants are found to be 79.05% for ResNet 10 8 A. Here, the GPU momory requirement is 4.57 GB during training and 4.02 GB during testing. It has 377.54 M parameters and required epoch time and batch time is about 17.10 seconds and 1.2214 seconds respectively. The model required about 32.62 second for testing and 1638 MB of system memory to store the trained model. *Figure* 12 (a) and (b) shows the plot of epoch vs train loss for UEDM dataset of different ResNet 10 architecture with shortcut A and B. *Figure* 12 (c) and (d) shows the plot of epoch vs test loss for UEDM dataset of different ResNet 10 architecture with shortcut A and B.

Table 3. Comparison table for the different ResNet 10 model architectures in terms of accuracy (A) in %, number of parameters (NP in millions), GPU memory required for training (GTR in GB), epoch time (ET in seconds), batch time (BT in seconds), GPU memory required for testing (GTS in GB), test time (TT in seconds) and model size (MS in MB) against each of the model using UEDM dataset.

Model	A	NP	GTR	ET	BT	GTS	TT	MS
ResNet 10 A	73.33	14.23	2.51	8.06	0.5757	1.69	19.20	**113.9**
ResNet 10 B	67.62	14.40	2.53	8.23	0.5879	1.71	18.75	115.3
ResNet 10_2 A	78.09	14.23	2.51	7.25	0.5179	1.69	19.59	113.9
ResNet 10_2 B	68.57	14.40	2.53	8.17	0.5836	1.71	19.15	115.3
ResNet 10_3 A	74.28	14.23	2.51	7.18	0.5129	1.69	19.69	113.9
ResNet 10_3 B	71.43	14.40	2.53	7.77	0.5550	1.71	19.35	115.3
ResNet 10_4 A	69.52	14.23	2.51	8.00	0.5714	1.69	20.03	113.9
ResNet 10_4 B	69.51	14.40	2.53	8.05	0.5750	1.71	19.87	115.3
ResNet 10_5 A	69.49	94.42	2.97	10.00	0.7143	2.46	23.47	472.3
ResNet 10_5 B	60.00	95.12	2.99	10.21	0.7293	2.48	22.77	475.8
ResNet 10_6 A	69.54	94.42	2.97	10.54	0.7529	2.46	23.17	472.3
ResNet 10_6 B	71.43	95.12	2.99	11.09	0.7921	2.48	22.72	475.8
ResNet 10_7 A	69.50	377.54	4.57	17.07	1.2193	4.02	32.38	1638
ResNet 10_7 B	66.67	380.34	4.61	17.97	1.2836	4.05	31.95	1659
ResNet 10_8 A	**79.05**	**377.54**	**4.57**	**17.10**	**1.2214**	**4.02**	**32.62**	**1638**
ResNet 10_8 B	71.43	380.34	4.61	17.64	1.2600	4.05	32.04	1659

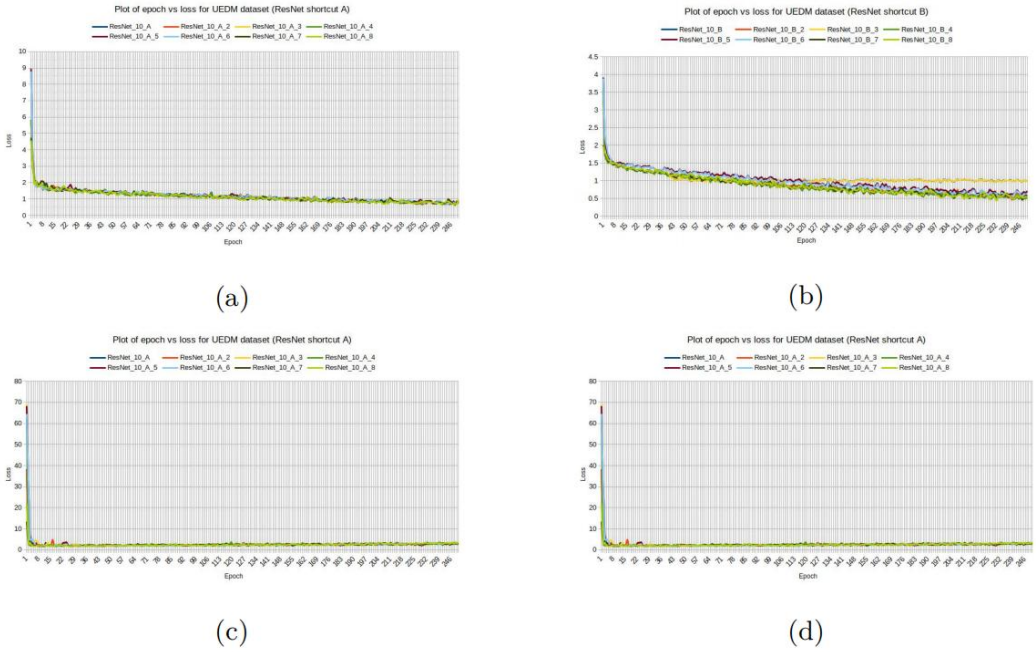

Figure 12: Plot of (a). epoch vs train loss for UEDM dataset with ResNet 10_A, (b). epoch vs train loss for UEDM dataset dataset with ResNet 10_B, (c). epoch vs test loss for UEDM dataset with ResNet 10_A and (d). epoch vs test loss for UEDM dataset with ResNet 10_B.

5.4. Overall Observations

The performances and other associated parameters for the best performing variants of ResNet 10 for three different dataset are recapitulated in *Table* 4. It may be observed from table 4 that ResNet 10_7_A provide better result (as compared to other model that we have tested) for UU2 and UCF Crime 13 dataset; which is about 98.66% and 53.68%. On the other hand, ResNet 10_8_A provide better result as compared to other models which is about 79.05%. The better resulting model have quietly larger amount of parameters, epoch time, batch time, GTR, GTS, test time and model size. From the point of best resulting model, classification performance plays major role as compared to other related parameters/ points. So, classification performance is superior to other points because lots of high computing resources are available in current days.

Table 4. Observation table for the best performing ResNet 10 network in terms of accuracy (A) in %, number of parameters (NP in millions), GPU memory required for training (GTR in GB), epoch time (ET in seconds), batch time (BT in seconds), GPU memory required for testing (GTS in GB), test time (TT in seconds) and model size (MS in MB) against each of the dataset.

Dataset	Model	A	NP	GTR	ET	BT	GTS	TT	MS
UU2	ResNet 10_7 A	98.66	377.54	4.57	33.35	0.5131	4.02	178	1638
UCF Crime 13	ResNet 10_7 A	53.68	377.54	4.57	27.09	1.1288	4.02	184	1638
UEDM	ResNet 10_8 A	79.05	377.54	4.57	17.10	1.2214	4.02	32.62	1638

6. CONCLUSION AND FUTURE DIRECTIONS

For the classification of different unusual human activity, we have to design such an algorithm which will work to catch up with those unusual events or activities so that those unusual events are automatically detected and criminals get caught in time. To work with this concept we have to design a learning based Artificial Intelligence system to meet the

requirements. We have to properly train the system for classifying usual as well as unusual activities that are performed by different human in surrounding environments. Here, we have used different structure of ResNet 10 model for our work. We have tested the above mentioned models with three different datasets namely UU2, UCF Crime 13 and UEDM and best accuracy are found to be 98.66% for UU2 dataset at ResNet 10_7_A, 53.68% for UCF Crime 13 dataset at ResNet 10_7_A and 79.05% for UEDM dataset at ResNet 10_8_A. Here, we have found less accuracy for the UCF Crime 13 dataset as compared to other datasets which is basically due to the improper augmentation of different classes in that dataset. In UCF Crime 13 dataset an another problem is that some of the video clips are overlapped in same videos as well as other videos. So in future we have to properly augment the dataset to get proper result of classification. In future we will try to increase the number of unusual classes in a activity dataset. Also, we will thinking to try to design to work with real time data so that a proper automatic visual surveillance system can be easily designed.

REFERENCES

[1] *J. A. Freer, B. J. Beggs, H. L. Fernandez-Canque, F. Chevriert and A. Goryashko, "Automatic recognition of suspicious activity for camera based security systems", Proceedings of the European Convention on Security and Detection,Brighton, UK, (1995) May 16-18.*

[2] *B. Maurin, O. Masoud and N. Papanikolopoulos, "Monitoring crowded traffic scenes", Proceedings of the IEEE 5th International Conference on Intelligent Transportation Systems, Singapore, (2002) September 6.*

[3] *V. Mahadevan, W. Li, V. Bhalodia and N. Vasconcelos, "Anomaly detection in crowded scenes", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA (2010) June 13-18.*

[4] *K. Soomro, A. R. Zamir and M. Shah, "UCF101: A Dataset of 101 Human Action Classes From Videos in The Wild", CRCV-TR-12-01, (2012) November.*

[5] *M. J. Roshtkhari and M. D. Levine, "Online Dominant and Anomalous Behavior Detection in Videos", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA (2013) June 23-28.*

[6] *F. B. Lung, M. H. Jaward and J. Parkkinen, "Spatio-temporal descriptor for abnormal human activity detection" Proceedings of the 14th IAPR Interenational Conference on Machine Vision Application (MVA), Tokyo, Japan (2015) May 18-22.*

[7] *K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA (2016) June 27-30.*

[8] *M. G. Narasimhan and S. S. Kamath, "Dynamic video anomaly detection and localization using sparse denoising autoencoders", Multimedia Tools and Applications., vol. 77, (2018), pp. 13173-13195.*

[9] *W. Sultani, C. Chen and M. Shah, "Real-World Anomaly Detection in Surveillance Videos", Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA (2018) June 18-23.*

[10] *T. Wang, Z. Miao, Y. Chen, Y. Zhou, G. Shan and H. Snoussi, "AED-Net: An Abnormal Event Detection Network", Engineering., vol. 5, no. 5, (2019) pp. 930-939.*

[11] *F. Dong, Y. Zhang and X. Nie, "Dual Discriminator Generative Adversarial Network for Video Anomaly Detection", IEEE Access., Vol. 8, (2020) pp. 88170-88176.*

[12] *H. Lv, C. Zhou, Z. Cui, C. Xu, Y. Li and J. Yang, "Localizing Anomalies From Weakly-Labeled Videos", IEEE Transactions Image Processing., vol. 30, (2021) pp. 4505-4515.*

[13] *T. Ahmed, M. S. Parvin, M. R. Haque and M. S. Uddin, "Lung Cancer Detection Using CT Image Based on 3D Convolutional Neural Network", Journal of Computer and Communication., vol. 8, no. 3, (2020) pp. 35-42.*

[14] *R. Cai, H. Zhang, W. Liu, S. Gao and Z. Hao, "Appearance-Motion Memory Consistency Network for Video Anomaly Detection", Proceedings of the AAAI-21/ IAAI-21/EAAI-21 Conference on Artificial Intelligence, Virtual, (2021) February 2-9.*

[15] *C. Hu, F. Wu, W. Wu, W. Qiu and S. Lai, "Normal Learning in Videos with Attention Prototype Network", arXiv preprint, 2021. https://doi.org/10.48550/arXiv.2108.11 055.*

[16] *M. I. Sarker, C. Losada-Gutirrez, M. Marrn-Romera, D. FuentesJimnez and S. Luengo-Snchez, "Semi-Supervised Anomaly Detection in Video-Surveillance Scenes in the Wild", Sensors., vol. 21, no. 12, (2021) pp. 3993-4012.*

[17] *W. Ullah, A. Ullah, T. Hussain, Z. A. Khan and S. W. Baik, "An Efficient Anomaly Recognition Framework Using an Attention Residual LSTM in Surveillance Videos", Sensors., vol. 21, no. 8, (2021) pp. 2811-2827.*

[18] *S. Habib, A. Hussain. W. Albattah, M. Islam, S. Khan, R. U. Khan and K. Khan, "Abnormal Activity Recognition from Surveillance Videos Using Convolutional Neural Network", Sensors., vol. 21, no. 24, (2021) pp. 8291-8306.*

[19] *L. Konwar, A. K. Talukdar, K. K. Sarma, N. Saikia and S. C. Rajbongshi, "Segmentation and selective feature extraction for human detection to the direction of action recognition", International Journal of Circuits, Systems and Signal Processing., vol. 15, (2021) pp. 1371-1386.*

[20] *A. Hussain, K. Muhammad, H. Ullah, A. Ullah, A. S. Imran, M. Y. Lee, S. Rho and M. Sajjad, "Anomaly based camera prioritization in large scale surveillance networks", Computers, Materials & Continua., vol. 70, no. 2, (2022) pp. 2171-2190.*

[21] *P. Suganthi, A. Jaganaath, J. Dhyanesh and A. Aravindan, "Suspicious Activity and Theft Detection Using Deep Learning", proceedings of the International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2024 May 9-10.*

[22] *J. Kukade and P. Panse, "Designing a Deep Learning Model for Video Anomaly Detection-Based Surveillance", Edited S. Fong, N. Dey and A. Joshi, ICT Analysis and Applications, ICT4SD 2023, Lecture Notes in Networks and Systems, Springer, Singapore. vol. 782 (2023), pp. 257-269.*

[23] *R. Nayak, U. C. Pati and S. K. Das, "A comprehensive review on deep learningbased methods for video anomaly detection", Image and Vision Computing., vol. 106, (2021) pp. 104078.*

[24] *M. Baradaran and R. Bergevin, A critical study on the recent deep learning based semi-supervised video anomaly detection methods. Multimedia Tools and Application., vol. 83, (2024) pp. 27761-27807.*

[25] *H.-T. Duong, V.-T. Le and V.T. Hoang, "Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey". Sensors., vol. 23, no. 11, (2023) pp. 5024-5048.*